REMARKS

Applicants appreciate the Examiner's time for participating in a telephone interview on March 10, 2004. In the interview, the Examiner disagreed with the applicants' assertion that cited references U.S. Patent 6,366,874 (Lee) and U.S. Patent 6,053,947 (Parson) do not disclose the underlying functional abilities of applicants' claimed invention pertaining to the use of Verilog and that Lee's graphical elements, although chosen by a designer, are not doing the same as applicants' invention. The Examiner disagreed at least for the reason of a difference in interpretation of claim language scope.

Applicants have amended claims 1, 4, 5, 9, 12-14, 18, 22, 24. No new matter has been added. Applicants respectfully request reexamination and reconsideration of claims 1-5, 7-9, 11-28.

The Examiner rejected claims 1-5, 7-9, 11, 14-23 under 35 U.S.C. 103(a) as being unpatentable over Lee in view of Parson. In the rejection, the Examiner states that Lee disclosed the applicants' claimed invention substantially as claimed, including a method for designing a logic circuit comprising:

> generating a functional design of a logic circuit by selecting, placing, and connecting reusable graphical library elements using a graphical user interface (GUI), the graphical library elements representing logical functions and connections between the logical functions;
>
> refining the functional design to represent a hardware design (functional design) of the logic circuit using the GUI;
>
> maintaining a data structure representative of a model, the model including combinational blocks, state elements, and graphical library elements of the logic circuit;
>
> the model including combinational blocks, state elements and graphical library elements of the logic circuit; and
>
> generating a simulation model of the functional design of the logic circuit and a separate (HDL) model of the hardware design of the logic circuit from the data structure [the cited portions are interpreted by examiner to read on the generation of separate models for the simulation and hardware because the views are disclosed by Lee to be user selected design objects which are separate and are models of the designers choice of components for further design processes].

Applicants respectfully disagree that Lee discloses the invention substantially as claimed by the applicants' amended claims. Claim 1, as amended, recites as follows.

> A method comprising:
> generating a functional design of a logic circuit by selecting, placing, and connecting graphical elements using a graphical user interface, the graphical elements representing elements of the logic circuit, the elements comprising state elements, combinatorial logic, reusable library elements, and elements containing textual description of logic;
> refining the functional design to represent a hardware design of the logic circuit using the graphical user interface to modify the connectivity of the elements;
> maintaining a data structure representative of the functional design; and
> generating an architectural, cycle-based simulation model of the functional design of the logic circuit and a separate implementation hardware description language (HDL) model of the hardware design of the logic circuit from the data structure, the implementation HDL model functionally equivalent to the architectural simulation model.

Lee does not disclose or suggest "generating a functional design of a logic circuit. . . using a graphical user interface" because Lee's teaching "resides in system and/or method for graphically browsing and exploring an electronic design, which is specified according to hardware description or similar language, such as verilog HDL or VHDL." (Lee, column 2, lines 55-58). As Lee states, "textual specification is often defined through source code written in one or more Hardware Description Languages (HDL)." (Lee, column 2, lines 14-16). Using textual specification to generate a functional design of a logic circuit is clearly distinguishable from using a graphical user interface to generate the functional design.

Lee does not disclose or suggest "generating an architectural simulation model of the functional design of the logic circuit. . ." As described in the applicants' specification (page 1, lines 4-12), "The architectural stage includes designing a framework of functional units that provide performance and functionality of a new microprocessor. This framework is typically captured in a text-based document. A model of the new microprocessor, represented in a high level language such as C++, is generated to verify that the function and performance requirements are met." Lee does disclose "selected presentations are referred to herein as "views," such that each view is generated dynamically and may involve one or more instances of user-selected design objects, (e.g., components, pins, circuits, logic blocks, interconnects, signal nets, buses, annotating notes, etc.)." (Lee, column 2, line 67 – column 3, line 5). Elsewhere, Lee

also discloses "design browser and/or explorer tool may generate one or more graphical representation or design view dynamically from HDL specification, preferably showing one or more level of design detail selected by user. Design browser and/or explorer tool may show user-selectively in design view, floating pins, net routing or other related interconnect or wiring design data." (Lee, column 5, line 63 – column 6, line 2). Lee's design browser and/or explorer tool with an ability to generate views of one or more levels of design detail does not teach or suggest generating an "an architectural simulation model of the functional design of the logic circuit".

Lee does teach "central to the design environment is HDL specification of system, circuit or prototype design under development. In this arrangement, various CAD tools, textual and/or graphical, which couple electronically or have data base access to the HDL specification, serve to capture, define, analyze, test, simulate, manipulate, or otherwise automatically or manually process portions or all of circuit being designed." (Lee, column 5, lines 27-35). However, this teaches away from "generating an architectural, cycle-based simulation model of the functional design of the logic circuit . . . from the data structure" since the applicants' data structure is representative of the functional design that was generated by "elements comprising state elements, combinatorial logic, reusable library elements. . ." in contrast to Lee's HDL specification. State elements, combinatorial logic, and reusable library elements are distinguishable from an HDL specification because an HDL specification does not include distinct state elements.

The Examiner states that Lee did not specifically generate a C++ (simulation model) and C++ classes, but Parson disclosed generating C++ and C++ classes.

Applicants respectfully disagree that the teachings of Lee and Parson can be combined to teach or suggest "generating a functional design of a logic circuit by selecting, placing, and connecting graphical elements using a graphical user interface . . . maintaining a data structure representative of the functional design . . . generating an architectural, cycle-based simulation model of the functional design of the logic circuit and an separate implementation hardware

description language (HDL) model of the hardware design of the logic circuit from the data structure."

Parson teaches "a computer-based simulation model that offers flexibility and efficiency in simulating a circuit while maintaining the same basic structure and notation as netlist languages. The programming convention used to express circuit design is similar in appearance and usage to hierarchical netlist languages, such as VHDL . . ." (Parson, column 2, lines 23-30). Parson also teaches "block 402 translates netlist and behavior model languages into the C++ convention of the present invention. The information source to this stage in block 401A is hierarchical netlist + behavior models. For purposes of discussion herein, assume that the former takes the form of VHDL structural information and the latter the form of VHDL behavior models." (Parson, column 13, lines 9-15). Parson also provides an example of the input block 401A with structural VHDL code in column 18, line 64-column 19, line 33.

Parson generates a simulation model that is event based. As one skilled in this art would know, an event-based simulation of a logic circuit is a simulation that calculates and updates changes in signals from blocks that are inputs to other blocks when a scheduler determines that the signals will be changed. As Parson's specification states on column 4, lines 2-7, "the simulation model uses a data flow convention to distribute signal values among model functions only on value changes, and only to those non-hierarchical models that use the signal values in their function." Parson's specification also states on column 5, lines 4-5, "The dataflow and scheduling features are utilized in simulation phase. According to the dataflow features, the process of simulating a circuit comprises distributing a signal only upon a change in the signal, and then only to those simulation models that use the signal."

An event-based simulation is distinguishable from a cycle-based simulation. As one skilled in this art would know, a cycle-based simulation is a simulation where signals are updated only once per cycle and gates are grouped in paths and the paths are scheduled as a group. When there are a lot of signal changes every cycle, a cycle-based simulation can be more computationally efficient because the cycle based simulation only updates signals once per cycle and the event-based simulation can update signals more than once per cycle.

The contrast between event-based and cycle-based simulation is provided by the applicants' specification on page 7, lines 17-21 as follows. "Hence the C++ model simulator is a cycle-based simulator. The Verilog model is also written after being extracted from the data structure and is typically simulated using an event driven simulator such as ModelSim from Model Technology, for example."

Parson teaches away from "generating an architectural, cycle-based simulation model of the functional design of the logic circuit and an separate implementation hardware description language (HDL) model of the hardware design of the logic circuit from the data structure" because Parson begins with HDL structural information and behavior models and generates an event-based simulation model. In contrast, the applicants' data structure represents a functional design of the logic circuit that was generated by state elements, combinatorial logic, and reusable library elements. The applicants' state elements are useful for generating the cycle-based simulation model because, as one skilled in this art would know, the state elements are processed differently than the non-state elements in a cycle-based simulation model. The applicants' originally filed specification states on page 7, lines 5-9 that "the process provides (116) timing and clock domain assignments partitions (118) clock domain topologies. Each partition is code ordered (120) and partition code provided to a C++ compiler. Code ordering means that the logical constructs are sorted based on producer/consumer. By subsequently code-ordering the C++ model may be simulated as a single call model. A single call model means that each logical construct is evaluated only once per cycle."

Parson also does not teach or suggest "generating . . . an separate implementation hardware description language (HDL) model of the hardware design, the implementation HDL model functionally equivalent to the architectural simulation model."

Accordingly, claim 1 is patentable over Lee in view of Parsons.

Applicant's claims 2-5, 7-8 are dependent upon, and further limit, claim 1. Accordingly, claims 2-5 are patentable over Lee in view of Parsons.

Applicant's claims 9, 14, 18, and 22, as amended, call for "generating a C++ architectural, cycle-based simulation model and a functionally equivalent implementation

Verilog model". Applicant submits that claims 9, 14, 18, and 22 are patentable over Lee in view of Parsons for at least the same reasons set out above with respect to claim 1.

Claim 9 is patentable over Lee in view of Parsons. Applicant's claim 11 is dependent upon, and further limits, claim 9. Accordingly, claim 11 is patentable over Lee in view of Parsons.

Claim 14 is patentable over Lee in view of Parsons. Applicant's claims 15-17 are dependent upon, and further limit, claim 14. Accordingly, claims 15-17 are patentable over Lee in view of Parsons.

Claim 18 is patentable over Lee in view of Parsons. Applicant's claims 19-21 are dependent upon, and further limit, claim 18. Accordingly, claims 19-21 are patentable over Lee in view of Parsons.

Claim 22 is patentable over Lee in view of Parsons. Applicant's claim 23 is dependent upon, and further limits, claim 22. Accordingly, claim 23 is patentable over Lee in view of Parsons.

The Examiner rejected claims 12 and 24 under 35 U.S.C. 103(a) as being unpatentable over Lee in view of Parson, and further in view of U.S. Patent 6,519,755 (Anderson).

Claim 9 is patentable over Lee in view of Parsons and further in view of Anderson. Applicant's claim 11 is dependent upon, and further limits, claim 9. Accordingly, claim 11 is patentable over Lee in view of Parsons and further in view of Anderson.

Claim 22 is patentable over Lee in view of Parsons and further in view of Anderson. Applicant's claim 24 is dependent upon, and further limits, claim 22. Accordingly, claim 24 is patentable over Lee in view of Parsons and further in view of Anderson.

The Examiner rejected claims 13 and 25 under 35 U.S.C. 103(a) as being unpatentable over Lee in view of Parson, and further in view of U.S. Publication 2002/0023256 (Seawright).

Claim 9 is patentable over Lee in view of Parsons and further in view of Seawright. Applicant's claim 13 is dependent upon, and further limits, claim 9. Accordingly, claim 13 is patentable over Lee in view of Parsons and further in view of Seawright.

Claim 22 is patentable over Lee in view of Parsons and further in view of Seawright. Applicant's claim 25 is dependent upon, and further limits, claim 22. Accordingly, claim 25 is patentable over Lee in view of Parsons and further in view of Seawright.

It is believed that all of the pending claims have been addressed. However, the absence of a reply to a specific rejection, issue or comment does not signify agreement with or concession of that rejection, issue or comment. In addition, because the arguments made above may not be exhaustive, there may be reasons for patentability of any or all pending claims (or other claims) that have not been expressed. Finally, nothing in this paper should be construed as an intent to concede any issue with regard to any claim, except as specifically stated in this paper, and the amendment of any claim does not necessarily signify concession of unpatentability of the claim prior to its amendment.

No fee is believed to be due. Please apply any charges or credits to deposit account 06-1050, referencing attorney/docket 10559-596001.

Respectfully submitted,

Date: April 8, 2004

Kenneth F. Kozik
Reg. No. 36,572

Attorneys for Intel Corporation
Fish & Richardson P.C.
225 Franklin Street
Boston, MA 02110-2804
Telephone: (617) 542-5070
Facsimile: (617) 542-8906